ApSoft-64

An Applesoft BASIC for the Commodore 64/128 computers

FS! Software

PO Box 7096
Minneapolis, MN 55407

Software written by Art Roberts
Produced by **FS! Software**

Manual written by Art Roberts and Don French

NOTICE OF REGISTERED TRADEMARK

Reference is made in several places in this manual to Commodore and Commodore 64 which are trademarks of Commodore Business Machines, Inc. References are made in other places in this manual to Apple and Applesoft which are registered trademarks of Apple Computer Corporation.

Dedicated to Wanda
Without whom this would not have been possible

TABLE OF CONTENTS

COPYRIGHT AND NOTICE OF LIMITATIONS OF WARRANTY AND LIABILITY

This Manual and the Apsoft-64 program on the accompanying floppy disk which is described by this manual are copyrighted and contain proprietary information belonging to FS! Software.

The manual may not be copied, translated, or reduced to machine readable form, in whole or in part, without the prior written consent of FS! Software.

The accompanying floppy disk may not be duplicated, in whole or in part for any purpose. Using unauthorized copies of the ApSoft-64 program may result in damage to the disk drive and should be avoided.

LIMITATIONS OF WARRANTY AND LIABILITY

FS! Software, or any dealer or distributor distributing this product, makes NO WARRANTY, EXPRESS OR IMPLIED, with respect to this manual or the related floppy disk, their quality, performance, merchantability, or fitness for any particular use. It is solely the purchaser's responsibility to determine their suitability for any particular purpose.

FS! Software will in no event be held liable for direct, indirect or incidental damages resulting from any defect or omission in this manual or the accompanying floppy disk, including but not limited to any interruption of services, loss of business or anticipatory profit, or other consequential damages.

THIS STATEMENT OF LIMITED LIABILITY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. FS! Software neither assumes not authorizes any other person to assume for it any warranty or liability in connection with the sale of its products.

INTRODUCTION

ApSoft-64 is a powerful extension to the resident Commodore 64 BASIC. It incorporates a 1541 program loader which speeds up loading of programs by a factor of 5. It also adds 45 easy-to-use-and-understand BASIC commands. These commands are in the same format as Applesoft BASIC for the Apple][family of computers. ApSoft-64 therefore gives the Commodore 64 owner a greatly expanded base of available software in addition to increasing his or her programming power.

Besides being able to write new programs using the advanced features of ApSoft-64, you will be able to use many hundreds of programs already written for Apple Computers. Several such programs have been provided free on the same disk as ApSoft-64. Included among these is the renowned "File Cabinet", the data base management program so popular among Apple owners.

Also, as a registered owner of ApSoft-64 you will be able to select from among hundreds of public-domain programs originally written for the Apple and recorded by us on Commodore disks. Priced at under $1 per program, you have but to return your completed warranty card to receive a catalog.

Also included on the ApSoft-64 disk is a terminal program to facilitate downloading Applesoft (and any other) programs via modem onto your Commodore 64's disk drive.

Note that disks designed to be used on Apple disk drives are physically incompatible with the 1541 disk drives, so Applesoft programs may not be loaded directly. Note also that not all programs written for the Apple are necessarily compatible with ApSoft-64. Only those which are written in Applesoft BASIC or integer BASIC are candidates. And among the Applesoft programs there are a few areas of incompatibility, detailed in the appendicies.

Fortunately, most programs will not require much, if any, modification. Those changes which do need to be made are usually minimal and can be accomplished by anyone with a bit of BASIC programming experience. To assist in the conversion process we have included a program on the disk which will analyze and convert an Applesoft text file, highlighting statements which might require your attention.

In addition to the Applesoft capability, several other commands have been added to give you even more programming power. Program control commands and commands to help you take advantage of the exciting sprite and music synthesis capability of your 64 have been included. Finally, ApSoft-64 has a built-in speedy loader which will load programs from the 1541 disk drive at up to five times the normal speed.

In sum, ApSoft-64 gives you the best of both worlds and then some. Welcome to the world of expanded horizons!

USING APSOFT-64

Apsoft-64 is an extension to the the Commodore-64's BASIC programming language. The commands it adds to the language are mostly from the Applesoft BASIC language as written for Apple][computers. In addition to Applesoft commands, several commands have been added which make programming the 64 easier and more fun. The superior graphics and sound synthesis capabilities of the 64 have also been made accessible.

When ApSoft-64 is loaded from the disk into the memory of the 64, it automatically replaces the BASIC language that comes with the machine. Like BASIC, it doesn't really do anything until you enter or load a program. Because ApSoft-64 is a superset of Commodore BASIC, you may load any existing BASIC program and expect it to run just like it always did.

You may expect it, but in a few cases it may not. ApSoft-64 needs the memory space between $C000 (49152) and $CFFF (53247). Although normal BASIC programs don't molest this space, any which do will need to be either modified or only run when ApSoft-64 is not present. This is also true of programs which are quite long as ApSoft-64 also uses the space from $9000 (36864) to $9FFF (40959). This block is at the end of normal BASIC's program space. As a result, the largest program which may be run under the ApSoft-64 operating system is 34814 bytes as compared to 38911 for normal BASIC. Fortunately, because of the more powerful command set, programs writtten with ApSoft-64 are more compact and require less memory.

Before looking at the commands of ApSoft-64 we should get it loaded into the computer.

Turn on the computer and the disk drive. Place the ApSoft-64 program disk into the drive.

Type: LOAD "APSOFT-64",8,1 [RETURN]

note: [RETURN] always means press the RETURN key. Also used in this manual are the <> brackets. These are used to bracket arguments required in ApSoft-64 BASIC commands. They are not intended to be actually typed in.

After a few seconds the copyright message will appear on the screen and a short time later ApSoft-64 will be loaded and ready for your command. The prompt character which is displayed is a "]".

Now on to the commands!

APSOFT-64 COMMANDS

The first commands to be examined are the disk control commands.

CATALOG

This command will display the directory of the disk currently in the drive. It will not destroy the program in memory. You may enter this command now to get a look at the contents of the ApSoft-64 disk. CATALOG may be used inside a program or in direct mode.

LOAD <program-name>

This LOAD command differs from the common BASIC LOAD in that it is not necessary to surround the program name in quotes nor is it necessary to add the ",8" after the name. The disk drive is assumed to be the device from which you want to load the program. The program name may not begin with a digit or quotes. All of the formats of LOAD commands valid with common BASIC are also valid with ApSoft-64. Although LOAD commands are allowed inside a BASIC program, the LOAD command must be the last command in the program line in which it occurs. Programs LOADed from inside a BASIC program will attempt to run as soon as they are loaded. If the program being loaded is longer than the program it is being loaded from, there is a high probability that the loaded program will cause an error which stops the execution of the program.

BLOAD <program-name>

BLOAD is functionally equivalent to LOAD"program-name",8,1. It may be executed from within an ApSoft-64 program without any bad effects. As always, the program name must be no more than sixteen characters long. You may specify the program name either as a string variable or as the explicit name. The following rules apply to explicitly stated program names:

The program name may not have a digit as its first character nor contain question marks. Nor may it contain a dollar sign after any character other than a space.

The BLOAD must be the last command on the program line in which it occurs.

RUN <program-name>

This is just like a LOAD and RUN combined. The disk drive is assumed to be the device from which to load the program. The same rules apply as with the LOAD command. RUN may be executed from within a BASIC program but must be the last command of the line. The normal formats used in common BASIC are also valid in ApSoft-64.

PR#8

This quaint little command, directly to you from Applesoft, is exactly like a RUN HELLO. The program with the name of HELLO will load from the disk and run for you. This command will work in either the direct mode or from inside a program. The program called 'HELLO' does not have to be the first program on the disk.

COMMANDS FOR PRINTING

PR#4

PR#4 will have the exact same result as OPEN 4,4:CMD4. It opens the printer as logical file 4 and causes all screen display commands (PRINT statements) to be re-directed to the printer. This command does not allow any secondary addresses to be sent to the printer. It may be used in either direct or program mode. If used in program mode, it will be automatically deactivated upon leaving the program.

PR#0

This command is used to reverse the effect of a PR#4. That is, it will have the same effect as typing in: PRINT#4:CLOSE4. This command can be used from either the direct mode or from inside a program.

GRAPHICS PROGRAMMING WITH APSOFT-64

The real power of ApSoft-64 lies in its graphics capabilities. The BASIC commands for graphics are in Applesoft format. Commands exist for switching painlessly into high-resolution graphics mode, to create split screens with both graphics and text, to draw figures on the screens, create sprites and various other graphic capabilities.

HIGH-RESOLUTION GRAPHICS

High-resolution screen display modes (hi-res) allow you to control each dot of the display. There are 320 dots across and 200 dots down the screen. The image of what is displayed on the screen is kept in the 64's memory. With ApSoft-64, you have two areas of memory set aside for hi-res screens. Each area contains 8192 bytes of memory. The "primary" hi-res screen area is at $4000-$5FFF (16384-24383). The secondary screen area is at $2000-$3FFF (8192-16191). It is important to understand that these screen areas fall right in the middle of what is normally ApSoft-64's program space. If proper precautions are not taken this conflict will cause your program to be wiped out by your hi-res commands.

PROTECTING THE GRAPHICS MEMORY

Like common BASIC programs, ApSoft-64 BASIC programs are stored in memory starting at $0800 (2048). The program itself is followed in memory by the storage space for the variables which are defined in the program. For example if a line of your program is "PI=3.14152", PI is a variable and the value 3.14152 is saved in PI's reserved place in variable storage. Numeric variables are stored starting at the end of the program as stated above but string storage starts at the top of BASIC's available memory space and is filled downward as strings are created or encountered in the program. The top of available memory storage is $94FF (38143) when

ApSoft-64 is started up. String variable storage can fill in all available memory in the variable storage space (from $8FFF down to the top of the program).

The question then is how can you prevent strings and other variables from destroying your hi-res screens and vice-versa. The answer is through two commands, HIMEM: and LOMEM: which define where variable storage should be located. LOMEM: sets the low address of variable storage and HIMEM: sets the high address. To reserve space for the primary hi-res screen and reserve $2500 (9472) bytes for variables, the combination of LOMEM:24576 and HIMEM:38144 could be used. (Actually, HIMEM: is at 38144 when ApSoft-64 is started up and does not need to be done again.) This puts variable storage at $6000-$9500 (above the hires screen). An interesting advantage of this is that if you save your program after creating a hi-res screen and giving the LOMEM: command, the hi-res screen will be saved along with your program and will be re-loaded right along with it.

Note that having hi-res screen capability does cut down the space available for your program. If you use only the primary screen, your program may not be longer than 14335 bytes ($3FFF-$0800) with 9472 additional bytes for variable storage. If you also use the secondary screen area, your program can extend from $0800-$1FFF (2048-8191) which is only 6143 bytes. Again this does not include the 9472 bytes of variable storage.

HIMEM: <address of top-of-BASIC>

This command will set the address of the top of BASIC. This is actually the highest address at which variables will be stored by the ApSoft-64 program. The maximum it can be is 36864, and the lowest is limited by the length of your program. (Programs begin at $0801 (2049) and extend upward).

LOMEM: <address of the beginning of variable storage>

This command will set the address of the top of your BASIC

program. It is the lowest address at which variables will be stored by the ApSoft-64 program. It is also the upper address of the area saved by a program save.

LOW-RESOLUTION GRAPHICS

There is an additional capability which accompanies the use of the primary hi-res screen area. You may effectively superimpose the hi-res graphics upon a low-resolution (lo-res) graphics screen. Several lo-res graphics commands are at your disposal for drawing and plotting on the lo-res background. The size of each lo-res "dot" is an 8 x 8 block of hi-res dots. There are therefore 25 rows of 40 dots available for lo-res graphics. Several commands are included in ApSoft-64 to assist in lo-res graphics plotting.

The background area of hi-res screens (on which the lo-res plotting commands operate) uses the same memory as the text screens. Lo-res plotting on the primary hi-res screen uses the secondary text screen and lo-res plotting on the secondary hi-res screen uses the primary text screen. This means that after displaying a primary hi-res screen, switching to secondary text mode will display a screen of garbage until it is cleared. The same is true of going from a secondary hi-res screen to a primary text screen.

The hi-res command HCOLOR= will have the effect of clearing the lo-res screen, and setting the color of the lo-res screen to black (default) if no color is specified.

GR

This command is functionally equivalent to the HGR command. In Applesoft BASIC it would cause a shift to a special mid-res mode not supported in ApSoft-64.

PLOT <X position (0 to 39)>,<Y position (0 to 24)>

This will plot a character-sized graphic block of the current character color on the currently displayed screen.

VLIN <starting Y position (0 to 24)>, <ending Y position (0 to 24)> AT <X position (0 to 39)>

This command will plot a vertical line on the lo-res screen.

The first pair of numbers specify the starting and ending of the line to be drawn and the number after the AT specifies the column in which it will be drawn.

HLIN <starting Y position (0 to 39)>, <ending Y position (0 to 39)> AT <X position (0 to 39)>

This command will plot a horizontal line on the lo-res screen. The first pair of numbers specify the starting and ending X-positions of the line to be drawn and the number after the AT specifies the row in which it will be drawn.

SECONDARY TEXT SCREEN

There is also a second text screen area. This means that you may at any time simply switch to the secondary text screen and whatever screen display you have previously created for the secondary text screen will pop onto the screen without having to re-issue the print statements which created it in the first place. You may just as easily switch back to the primary screen and have its previous contents be displayed. Certain preparations must be made before writing to the secondary screen, however.

The text screen, like hi-res screens, need to have a block of memory reserved for storing the character information which gets displayed on the screen. The text information which is associated with the secondary text screen is at $6000-$63FF (24576-25576). So, to protect it you must set LOMEM: at $6400 (25600) instead of $6000

as described above. You must also load a character set from disk into the proper memory space, $6800-$6FFF (26624-28672). We have included two character set files on the disk, C64.CHARSET and MACH.CHARSET, for this purpose. The BLOAD command is ideally suited for loading character sets.

Finally, you must tell ApSoft-64 that you want the secondary text screen to be the one you will use for subsequent print statements. This is accomplished through the use of the curious Applesoft technique known as "soft-switches", explained more fully further on. In short, a "POKE -16299,1" will switch the program into secondary text screen mode and a "POKE-16300,1" will switch it back to the primary screen. If the screen is in hi-res graphics mode when you want to switch to the secondary text screen, be sure to switch to text mode before switching to the secondary screen. Split-screen mode must also be cleared before switching to the secondary screen. Hi-res and split-screen capabilities are described in the following paragraphs.

There is a program on the disk, PG2 CHAR DEMO, which demonstrates using the secondary text screen.

USING THE HI-RES GRAPHICS SCREENS

The backgrounds of hi-res screens are composed of 25 rows of 40 8-by-8 hi-res dots. Each block of dots may have its color defined independently of the rest of the screen. The background color information is stored in an area of accessible memory. In fact, the primary hi-res screen's background memory coincides with the secondary text screen memory and the secondary hi-res screen's background memory is coincident with the primary text screen memory. This means that switching to primary hi-res mode when a secondary text screen remains uncleared will result in a hi-res display with a background of varied colors. As this is not what is generally desired, the conclusion is that a clean primary hi-res screen may not co-exist with a secondary text screen and a clean secondary hi-res screen may not co-exist with a primary text screen. Of

course, you may clear the offending text screen before going into hi-res mode if you wish but the text screen display will not be resurectable without re-executing the PRINT statements which created it in the first place.

SPLIT SCREEN MODE

Besides the text, hi-res, and combination modes, there is also a split screen mode of display. This mode is possible with both the primary and secondary hi-res screens. It reserves the bottom five lines of the hi-res screen for text display only. Your program may issue PRINT commands while the hi-res screen is displayed and the information will be displayed in the five-line window below the hi-res display area. Split screen mode is activated via the soft-switches mentioned above. A more complete discussion of the soft-switches follows.

THE SOFT-SWITCHES

Applesoft has a unique way of controlling the selection of display modes. ApSoft-64 has duplicated this system. It involves PEEKing or POKEing an address specified in the range of -16229 to -16304. Yes, those are negative addresses! The value POKEd is not relevant and a PEEK works as well as a POKE. Only the addresses referenced are significant. The effect of using these commands is to toggle a switch which tells ApSoft-64 what mode of display we want to be in. It causes a switching between modes without erasing what is currently being displayed. That is, you may return to a mode you switched out of, and, as long as the screen memory was not modified in the meantime, you will find the same information displayed on the screen as when you left.

POKE/PEEK ADDRESS	EFFECT
-16299	Switch to secondary screen
-16300	Switch to primary screen
-16302	Switch split-screen off

-16301.....Switch split-screen on
-16303.....Switch hi-res graphics off
-16304.....Switch hi-res graphics on

SECONDARY SCREEN SWITCH (-16299)

POKEing or PEEKing this address will switch to the secondary display screen. It switches to the secondary hires screen when in hires mode and to the secondary text screen when in text mode.

PRIMARY SCREEN SWITCH (-16300)

This functions like the one above except it causes the primary display screen to be selected.

SPLIT-SCREEN ON (-16301)

Turning this switch on puts the display in split-screen mode. This switch will only have an effect if the screen is already in graphics mode. Note that for the primary hi-res mode, selected by the HGR command, split-screen is automatically active until explicitly turned off. Turning on the secondary hi-res screen's split-screen mode requires following a specific sequence. Before entering secondary screen mode, split-screen must be turned off. Next, graphics must be turned on followed by selecting secondary screen mode followed finally by selecting split-screen mode.

SPLIT-SCREEN OFF (-16302)

Turning split-screen off only has effect when in graphics mode and when split-screen mode is on. Its effect is to make the entire screen a hi-res graphics display.

HI-RES GRAPHICS ON (-16304)

Turning hi-res graphics on will switch the display from text mode into graphics mode. Whatever text information is on the screen

will be replaced by the graphics screen. If primary mode is active, the primary graphics screen will be displayed. If the secondary mode is active, the secondary graphics screen will be displayed. Note that the soft-switch does not clear the graphics screen. An HGR or HGR2 will be necessary to accomplish that.

HI-RES GRAPHICS OFF (-16303)

Turning hi-res off returns the display to the text mode. If in primary mode, the primary text screen will be displayed. If in secondary mode, the secondary text screen will be displayed. Recall that secondary text mode requires the loading of a character set.

THE GRAPHICS COMMANDS ARE SUMMARIZED BELOW:

HGR

HGR has no arguments. It shifts the display mode to the primary hi-res graphics mode with split screen activated. The screen will clear upon execution of this command. This is the only way to clear the primary hi-res screen. It is usable in either direct or in program mode.

HGR2

Like HGR, HGR2 shifts the program into hi-res graphics mode. The secondary graphics display screen is selected and split-screen is not activated. The screen will be cleared by this command. It is not usable in the direct mode.

TEXT

Regardless of the status of any of the mode switches, TEXT will cause the primary text screen to be selected and split screen will be turned off. If the secondary text or graphics screen was active at the time of execution of this command, a CLR/HOME will

automatically be performed by TEXT. Lowercase mode will be selected if TEXT is executed in hi-res mode and upper case when executed in any other mode.

HCOLOR=<plot-color>,<background-color>

Both the color of dots being drawn and the background color are selected via the HCOLOR= command. The values of the background color may be in the range of 0-15 and the plot color in the range of 0-16, 16 having special significance. (See Appendix A for a list of all colors and their numbers.) A plot-color value of 16 causes the plotting command to be shifted into erase or "unplot", where dots plotted will be made the same color as the background. The "plot" mode may be re-established by another HCOLOR command specifying a plot color of 1-15. The default color of the background is black and the dots white upon starting ApSoft-64. Since all the 8-by-8-dot background blocks are cleared to the same color with this command, it has the effect of erasing any low-res designs which might appear behind the hi-res plots. The <,background-color> parameter is optional

An important difference between the Commodore 64 and the Apple is the inability of the 64 to select various colors for each discrete pixel on the screen. In hi-res mode, it is limited to one background and one foreground color in each 8 by 8 block of pixels. Certain compromises had to be made to accomodate these differences. One of these involves the way the HCOLOR command functions. In ApSoft-64, HCOLOR sets all 8 by 8 blocks to the same foreground and background colors. In Applesoft, the HCOLOR command sets the color of the dots about to be plotted only, leaving the previously plotted dots in their original color configurations. The usual way of erasing a dot in Applesoft is to HPLOT the dot with the same color as the background color.

SET <x>,<y>,<background-color>,<dot-color>

Individual 8-by-8 blocks may have their background and dot

colors set with SET. The X and Y values may range from 0-39 and 0-24 respectively.

HPLOT <x1>,<y1> TO <x2>,<y2> TO <x3>,<y3> TO...

It is with this command that plotting points and lines on the hi-res screens is accomplished. The x and y values may range from 0-319 and 0-199 respectively. The upper left corner of the screen is 0,0 and the bottom right is 319,199. If only one set of coordinates is specified a point will be drawn. Otherwise, lines are drawn between the specified points. Plotting is always done on the hi-res screen last displayed.

SHAPE TABLES

Apsoft-64 allows you to create and save line drawings (such as character sets). You may later recall and draw the shapes on hi-res screens. These designs are called shapes and up to 256 shapes may be grouped into what is called a shape table. Shapes are drawn on the screen with the DRAW command and erased with the XDRAW command. Shape tables may be created with a special shape-editor program, SHAPE-CREATE, provided on the ApSoft-64 disk, or they may be created "by-hand". Appendix F provides the system information for building your own shape tables. The tables are loaded from disk back into the 64 with the SHLOAD command.

SHLOAD <shape-table-name>

The shape-table-name is that which you stored the shape table under on the disk. The SHLOAD will load the shape table into the same memory space from which it was saved. There is a two-byte address pointer at 53016/53017 which tells ApSoft-64 where the shape table is in memory. The SHLOAD command automatically sets this pointer correctly upon loading the shape.

DRAW <shape#> AT <x>,<y>

Shape# specifies which shape within the table of shapes to draw on the screen and x and y specify at which position on the screen to start drawing the shape. The x and y coordinates refer to the pixel number in the horizontal and vertical directions respectively. The x-pixels go from 0 to 319 from left to right and the y-pixels go from 0 to 199 from top to bottom.

XDRAW <shape#> AT <x>,<y>

XDRAW is the command for erasing shapes from the screen and the format is identical to the DRAW command.

SCALE= <magnification of shape (0 to 255)>

This command will cause the DRAW and XDRAW command to magnify each subsequent shape DRAWN or XDRAWN until the next SCALE command is issued.

ROT= <relative rotation>

This command is used to rotate the shapes drawn by increments of 45 degrees, regardless of the scale. Zero is no rotation, 8 is a 45 degree rotation, 16 is 90 degrees, 24 is 135 degrees, 32 is 180 degrees, 40 is 225 degrees, 48 is 270 degrees, and 56 is 315 degrees. ROT effects all DRAWings of shapes until another ROT is issued.

SPRITE COMMANDS

The Commodore-64 supports movable screen shapes called sprites. These are different than the line-drawings called shapes in Applesoft. They differ in that any of up to eight sprites may be independently moved and expanded or contracted in both the

horizontal and vertical directions. ApSoft-64 makes it much easier to turn sprites on and off and to move them about the screen with a few simple commands.

SPRITE <sprite#>,<block>,<X expanded>,<Y expanded>

This command activates a sprite which has already been built. Sprites must be created in the normal Commodore way, preferrably with a sprite editor. A public domain sprite editor has been included on the disk. Each sprite takes up 64 bytes of memory. The location of this 64-byte block must be within a single 16k block of memory. Sprites which will appear on the primary hi-res or seconday text screen must be defined within the 16k area which starts at address 16384. The 16k block which starts at 0 must be used for the secondary hi-res or primary text screen. The <block> parameter which must be specified in the SPRITE command may range from 0 to 255 and identifies which 64-byte block within the 16k area has been used to define the sprite being activated. ApSoft-64 will automatically select the proper 16k block based on which mode is active when the SPRITE command is issued. You must, therefore, know the address of where your sprites are set up in order to use the SPRITE command. Two equations are used for translating between sprite start address and <block> number: BL=(AD-16384)/64 and BL=AD/64. The first is for sprites in the second 16 block and the second is for those in the first block.

<X expanded> is 0 if the sprite is not to be expanded in the horizontal direction and 1 if it is. <Y expanded> follows the same format for the vertical expansion choices.

SPRT1 <sprite#>,<extended color mode>,<color>, <ext. color 1>,<ext. color 2>

This command is used to set the colors of the sprite, and used to turn on the extended color mode. <extended color mode> is 0 for off and 1 for on, <color> is the color of the sprite in unextended mode, <ext. color 1> is the extended color 1, and <ext. color 2> is

the extended color 2.

MOVE <sprite#>,<X position>,<Y position>,<0=off/1=on>

This command is used to move the sprite around on the screen, and to turn the sprite on or off. Turning the sprite off does not effect the video chip registers, as the sprite is still defined and can be again displayed by using the MOVE command and turning the sprite on again.

HOME

This command is the same as the following command in Commodore 64 BASIC:

PRINT "<shift/clr-home>"; <return>

INVERSE

This command effectively prints a reverse-on character prior to every print statement executed, causing all screen output to be printed in reverse. INVERSE mode remains in effect until a NORMAL command is issued.

NORMAL

This command will cause a reverse-off character to be printed prior to any print statement, causing all screen output to be in normal (non-reverse).

SPEED= (a number from 0 to 255) <return>

This will provide a delay in printing each character; where 0 is the slowest print speed and 255 is the fastest.

COLOR=

This will set the color of the characters printed on the screen according to the color table listed in the appendix. See also Graphics commands.

VTAB <line number (0 to 24)>

This command will move the cursor to the first character of the screen line specified. VTAB must be used prior to HTAB.

HTAB <character position (0 to 39)>

This command will set the cursor to the character position specified on the line where the cursor is located. If HTAB is to be used in conjunction with VTAB, VTAB must be used first.

GENERAL SYSTEM COMMANDS

DEL <starting line>,<ending line>

This command will either delete a single BASIC line, a range of BASIC lines, or all BASIC lines. If starting and ending line numbers are not specified, all lines of the program will be deleted. If starting line number is omitted, all lines up to the ending line number will be deleted. If the ending line number is omitted, all lines from the starting line number to the end of the program will be deleted.

TRACE

This command will, when a program is running, list each line as it is executed, printing:

LINE#<BASIC line # here>

NOTRACE

This command will turn off trace mode.

POP

POP is a way of canceling a subroutine without returning to the GOSUB point. It puts the program into the subroutine level one higher than it was at the time the POP was executed. If a POP is executed in a subroutine within a subroutine, the next RETURN executed before another GOSUB will cause a return to the first GOSUB rather than the second one.

ONERR GOTO <line #>

This command is very useful in error trapping, as it will specify which line to goto when an error occurs when a BASIC program is running. If the line # specified is not in the program, or the ONERR is not followed by a GOTO, an additional error may occur resulting in some confusion for the program, and the results will be unpredictable. Location 53040 contains the error number the program encountered. The Appendicies contain a list of numbers and associated errors. The ONERR command must be in the program and executed prior to any errors or a normal error exit from the program will occur. The ONERR command will not recognize errors generated by the disk drive.

RESUME

This command will cause execution to return to the line AFTER the line where the error occured that caused ONERR to execute.

CALL

This is the same command as the Commodore SYS command. It was included to maintain compatibility with Applesoft. One area of incompatibility, however, is the inability of ApSoft-64 to handle CALLs to negative addresses.

CLEAR

This is the same as the Commodore CLR command. It will clear out all variables in the BASIC program in memory.

IN

This command is to be used to restore the ApSoft-64 following an interruption of the program via a press of <run-stop/restore>. It will not erase the program in memory, but will reset pointers and the ApSoft-64 system, and restore the correct prompt. Your program may run following such an interruption without typing IN, but some commands may give errors and others not be available. The IN command is sometimes uesful when in a hi-res screen mode. Even though you may not be able to see the letters as you type them, the command will have the desired effect of bringing you back to text mode.

PDL <number of paddle (0 to 3)>

This is a function command and will return a value ranging from 0 to 255, corresponding to the rotational position of the paddle. Paddle numbers 2 and 3 are the normal Commodore paddles plugged into port 1. Paddle numbers 0 and 1 are used for joysticks plugged into ports 1 or two, respectively. See the appendicies for the values returned if the joysticks are used. PDL commands, (and the PEN commands) must be enclosed in parentheses when used as a term in an expression:

X=(PDL(0))*25

PENX

This function command returns the horizontal screen position of a light pen that is plugged into port 1. In order to be used in an expression, it must be enclosed in parentheses. The value returned is from 50 to 250 and represents pixel position 0 to 199 (with a resolution of 1 pixel). To convert PENY to the nearest vertical pixel position, use the equation PY=(PENY)-50.

PENY

This function command returns the vertical screen position of a light pen that is plugged into port 1. In order to be used in an expression, it must be enclosed in parentheses. The value returned is from 25 to 125 and represents pixel position 0 to 199 (with a resolution of 2 pixels). To convert PENY to the nearest vertical pixel position, use the equation: PY=(PENY)*2-50

SOUND <voice#>,<waveform>,<pulse width (if pulse selected)>,<attack>,<decay>,<sustain>,<release>

This command is used to set up the sound registers with the voice, waveform, pulse width, and ASDR values being the parameters needed. It is used in conjunction with the command PLAY, which specifies the voice and frequency and starts the sound playing, and the command OFF, which turns the specified voice off.

Waveform is as follows: 16 = triangle, 32 = sawtooth, 64 = pulse, 128 = noise. The pulse width, if pulse is used, is a value from 0 through 4096. The attack, decay, sustain and release each are values from 0 through 15.

PLAY <voice#>,<frequency>

This is the command that determines the frequency of the sound

played, and turns on the gate bit that starts the sound being played. If the OFF command is not used, the value of the sustain parameter sets the level the sound will stay at.

The frequency is a value from 0 through 65535. See the appendix for a table of values and notes.

OFF <voice#>

OFF is a command that will turn off the gate bit and start the release phase of the sound.

APPENDIX A

COLOR TABLE

C-64		APPLE	
NUMBER	COLOR	LO-RES	HI-RES
0	BLACK	BLACK	BLACK
1	WHITE	MAGENTA	GREEN
2	RED	DARK BLUE	VIOLET
3	CYAN	PURPLE	WHITE 1
4	PURPLE	DARK GREEN	BLACK 2
5	GREEN	GREY	ORANGE
6	BLUE	MEDIUM BLUE	BLUE
7	YELLOW	LIGHT BLUE	WHITE 2
8	ORANGE	BROWN	-----
9	BROWN	ORANGE	-----
10	LIGHT RED	GREY	-----
11	GRAY 1	PINK	-----
12	GRAY 2	GREEN	-----
13	LIGHT GREEN	YELLOW	-----
14	LIGHT BLUE	AQUA	-----
15	GRAY 3	WHITE	-----

APPENDIX B

ERROR TABLE FOR ONERR GOTO

1	TOO MANY FILES
2	FILE OPEN
3	FILE NOT OPEN
4	FILE NOT FOUND
5	DEVICE NOT PRESENT
6	NOT INPUT FILE
7	NOT OUTPUT FILE
8	MISSING FILE NAME
9	ILLEGAL DEVICE NUMBER
10	NEXT WITHOUT FOR
11	SYNTAX
12	RETURN WITHOUT GOSUB
13	OUT OF DATA
14	ILLEGAL QUANTITY
15	OVERFLOW
16	OUT OF MEMORY
17	UNDEF'D STATEMENT
18	BAD SUBSCRIPT
19	REDIM'D ARRAY
20	DIVISION BY ZERO
21	ILLEGAL DIRECT
22	TYPE MISMATCH
23	STRING TOO LONG
24	FILE DATA
25	FORMULA TOO COMPLEX
26	CAN'T CONTINUE
27	UNDEF'D FUNCTION
28	VERIFY
29	LOAD
30	BREAK

Please refer to your User's Manual or Programers Reference Guide for an explanation of these errors.

APPENDIX C

SOUND VALUES

NOTE	OCTAVE							
	0	1	2	3	4	5	6	7
C	268	536	1072	2145	4291	8583	17167	34334
C#	284	568	1136	2145	4547	9094	18188	36376
D	301	602	1204	2408	4817	9634	19269	38539
D#	318	637	1275	2551	5103	10207	20415	40830
E	337	675	1351	2703	5407	10814	21629	43258
F	358	716	1432	2864	5728	11457	22915	45830
F#	379	758	1517	3034	6069	12139	24278	48556
G	401	803	1607	3215	6430	12860	25721	51443
G#	425	851	1703	3406	6812	13625	27251	54502
A	451	902	1804	3608	7217	14435	28871	57743
A#	477	955	1911	3823	7647	15294	30588	61176
B	506	1012	2025	4050	8101	16203	32407	64814

These values are for the equal-tempered scale where the note A in octave 4 is 440 hz (concert pitch).

APPENDIX D

MEMORY MAP

E000 - FFFF COMMODORE'S OPERATING SYSTEM
(57344 - 65535)

D000 - DFFF INPUT OUTPUT DEVICES AND COLOR RAM
(53248 - 57343)

A000 - BFFF COMMODORE'S BASIC OPERATING SYSTEM
(40960 - 49151)

C000 - CFFF ApSoft-64 OPERATING SYSTEM
(49152 - 53247)

9000 - 9FFF ApSoft-64 OPERATING SYSTEM
(36864 - 40959)

0800 - 8FFF BASIC PROGRAM STORAGE
(2048 - 36863)

6800 - 6FFF SECONDARY TEXT PAGE CHARACTER SET (MUST LOADED IN RAM)
(26624 - 28671)

6000 - 63FF SECONDARY TEXT PAGE / PRIMARY HIRES BACKGROUND PAGE
(24576 - 25599)

6400 - 67FF COLOR MEMORY SAVE AREA
(25600 - 26623)

4000 - 5FFF PRIMARY HIRES PAGE
(16384 - 24575)

2000 - 3FFF SECONDARY HIRES PAGE
(8192 - 16383)

0400 - 07FF PRIMARY TEXT PAGE / SECONDARY HIRES BACKGROUND
(1024 - 2047)

0000 - 3FFF FIRST VIC II MEMORY BLOCK
(0 - 16383)

4000 - 7FFF SECOND VIC II MEMORY BLOCK
(16384 - 32767)

8000 - BFFF THIRD VIC II MEMORY BLOCK
(32768 - 49151)

C000 - FFFF FOURTH VIC II MEMORY BLOCK
(49152 - 65535)

APPENDIX E

RESERVED WORDS

The following words are 'tokenized': each word is turned into a one byte token to be used by the operating system. If any of these words are used as variables an error will occur.

ABS	AND	ASC	AT	ATN	
CALL	CHR$	CLEAR	COLOR=	CONT	COS
CMD	CLR	CATALOG			
DATA	DEFF	DEL	DIM	DRAW	
END	EXP				
FN	FOR	FRE			
GET	GET#	GOSUB	GOTO	GR	GO
HCOLOR=	HGR	HGR2	HIMEM:	HLIN	HOME
HPLOT	HTAB				
IF	INPUT	INT	INVERSE	INPUT#	IN
LEFT$	LEN	LET	LIST	LOAD	
LOG	LOMEM:				
MID$	MOVE				
NEW	NEXT	NORMAL	NOT	NOTRACE	
ON	ONERR	OR	OPEN		
PDL	PEEK	PLOT	POKE	POP	POS
PRINT	PRINT#	PR#	PEN	PLAY	
READ	REM	RESTORE	RESUME	RETURN	RIGHT$
RND	ROT=	RUN			
SAVE	SCALE=	SGN	SHLOAD	SIN	SPC(
SPEED=	SQR	STEP	STOP	STR$	SPRITE
SPRT1	SOUND	SYS	STATUS		
TAB(	TAN	TEXT	THEN	TIME	
TIME$	TO	TRACE	TI$	TI	
USR					
VAL	VERIFY	VLIN	VTAB		
WAIT					
XDRAW					

APPENDIX F

UNDERSTANDING SHAPE TABLES

ApSoft-64 will allow the use of all the Applesoft commands for the use of shape tables. This includes an enhancement that allows the use of SHLOAD to load a shape table from the disk drive, and set the pointers. An Applesoft shape table can be typed in from magazine listings, for example, and be used with no changes, other than paying attention to the memory maps to avoid conflicts with any memory used by the system. In general, the best location for a shape table is at the top of memory with HIMEM: command used to protect the shape table from Basic.

The program SHAPE CREATE on the ApSoft-64 disk will assist in making shape tables. It will allow the creation of a shape table with up to 255 shapes in it, each having up to 255 vectors. It has options for both LOADing and SAVEing shape tables, and has the ability to display a shape on the screen after editing it.

The address pointer to the location of the shape table is at locations 53016 and 53017. It will be set automatically with the SHLOAD command when a shape table is loaded.

The first byte of the shape table is the number of shapes in the table (in hex), the next byte is a zero byte. Following the zero byte is a series of bytes that are pointers to the actual shape defintions. Each set of pointers is in the standard high order/lo order format, and is the offset from the starting byte of the table to the shape definition it refers to.

The actual shape definitions follow all the pointers and are seperated by zero bytes. The definitions are like a series of commands to a plotting machine. Each byte of the definition list contains three commands. The first two can either plot and move or just move, and the last can only move.

```
          3rd..   2nd .....   1st .....
BIT:      7   6   5   4   3   2   1   0
CONTENTS: M   M   P   M   M   P   M   M
```

Where each vector is a two bit representation of a direction:

00 Move up;
01 Move right;
10 Move down;
11 Move left.

The P is 1 to plot and 0 to just move.

Also keep in mind that if the 3rd vector is all zeros then it will be ignored; likewise, if the 2nd AND the 3rd vectors are all zeros both will be ignored. If the 2nd vector contains zeros and the third does not then all three will be used.

The basic procedure for the design of a shape table is to lay out the design on graph paper one dot (or vector) at a time, and when all the vectors are laid out, then lay the vectors out in a straight line. The vectors are then written down two or three at a time (the third being used when it doesn't need to plot), and the ones and zeros are then put in order for each byte.

In designing shapes, it is helpful to use graphing paper to lay out your designs, and to start the shape in the center of the shape so when the ROT command is used, the shape will rotate about its center. The next step in designing the shape is to unwrap the vectors and lay them out left to right, then divide them into groups of two or three (remembering how the vectors are put together). Next, take the groups of vectors and convert the vectors to hex.

Since up to 255 shapes can be placed into a table, each shape in the table should be designed first, then the shapes should be placed in memory just above the number of pointers needed for the

table being designed. Each set of pointers should be calculated and place into the table, remembering that a zero byte is placed at the end of a particular shape to indicate the end of that shape. The program SHAPE CREATE will be most helpful in creating shapes.

APPENDIX G

JOYSTICK VALUES FOR PDL (0) AND PDL (1)

VALUE	MEANING
31	NOTHING
15	FIRE BUTTON PUSHED
14	FIRE BUTTON & UP
13	FIRE BUTTON & DOWN
11	FIRE BUTTON & LEFT
7	FIRE BUTTON & RIGHT
6	FIRE BUTTON & UP & RIGHT
10	FIRE BUTTON & UP & LEFT
5	FIRE BUTTON & DOWN & RIGHT
9	FIRE BUTTON & DOWN & LEFT
30	UP
29	DOWN
27	LEFT
23	RIGHT
22	UP & RIGHT
26	UP & LEFT
21	DOWN & RIGHT
25	DOWN & LEFT

APPENDIX H

FILE CABINET PROGRAM

This public domain Applesoft program is a database manager. It will allow you to set up a data base, define the different fields of information with titles; enter, change, search for data; sort data and print out reports.

Prior to using FILE CABINET it is recommended that a new disk be formated and FILE CABINET saved on it. Since this is a public domain program, it can be placed on every data disk that is used, and will save your ApSoft-64 disk from undue wear and accidental erasures.

When first run, FILE CABINET will check for database names, and if none are found, the first menu screen will give the choices:

```
1 QUIT
2 CREATE A NEW DATA BASE
```

If any databases have already been defined, they will show up as the first choices on the screen. Selecting choice # 2 will result in the question:

```
NAME FOR NEW DATA BASE?
```

The name for the new database should not be over 8 characters - or even less if reports are to be saved. One suggestion is to use only 7 characters for the name of the database and a letter for any reports that are to be saved.

Upon hitting return, the question will be:

```
HEADER FOR COLUMN NUMBER 1?
```

This question is the first of several that sets up the fields of data that are to be used. For example: the name of the database could be MAILIST; and the headers could be:

```
HEADER #1:NAME
HEADER #2:ADDRESS
HEADER #3:CITY
HEADER #4:STATE
HEADER #5:ZIP
HEADER #6:PHONE #
```

Upon hitting return, the main menu will appear. If no headers are input, the first menu will appear.

The main menu will display the name of the current database, the number of records currently in that database, and the total number of records allowed in that database. In addition, the printer status will be displayed, either ON or OFF.

The main menu will allow 9 choices:

1. This will cause the first menu to appear so that either a new database can be created or another selected.

2. This allows the entering of the data to each of the fields.

3. This will allow you to search the database for records containing specific data. You may search for all records containing a desired string of characters. All records beginning with the indicated string in the indicated field will be found. That is, if the name ANDERSON is being searched for, the names ANDERSON BILL, ANDERSON JOE AND ANDERSON JOHN will all be found and listed.

4. This choice will delete selected records by record number.

5. This will cause the program to search on the disk for any report formats previously saved. If none are found, the choices

will be either to return to the main menu or to create a report format. To create a report format, the total number of headers to be used must be chosen. The choices for header will be displayed on the screen and will include the record number as a header choice. Now the header will be chosen, along with the tab position. If there are any numeric columns, they may be flagged to be summed vertically, and additional choice of adding up the rows (horizontally) will be given. After completing this, the date can be entered, and a choice of ALL files is given. Choosing none will cause a set of choices that allows selected data to be printed based on one or more headers. When the report is finished printing, choices will allow the saving of the report format and another printing of the report.

6. This choice will allow the sorting of the records by a single field, using either alphabetic and numeric sorting or descending numerical sorting.

7. This will cause the listing of all the data fields.

8. This will toggle the printer: on if it is off and off if it is on.

9. This will exit the database.

Please note that the Filecabinet program is presently set up for use with the Commodore 1525 printer. The lines to change to be able to use this program with another printer are the subroutines starting at lines 2000 and 55000. The subroutine at 2000 turns on the printer (it does a PR#4), and if a wider report is to be used the proper print command for your printer to set up a wide page should be placed here. The subroutine starting at line 55000 sets up a tab. The variable ZQ is the tab position needed, and the string 'ZQ$' should contain the control sequence for your printer to tab to the correct position.

APPENDIX I

PROGRAMS ON ApSoft-64 DISK

APSOFT-64

This is the program that loads ApSoft-64.

HELLO

Hello is a program that will act as a 'menu', and can be loaded and run by a PR#8.

SHAPE DEMO

This loads the file SHAPES $8000 using the SHLOAD command and demonstrates the use of shapes on the hi-res screen.

PG2 CHAR DEMO

This program loads the two files C64 CHARSET and MACH.CHARSET, (which are character sets), and demonstrates the use of the secondary text page.

ARTSPACE

This is an actual Apple program modified to use a shape, and demonstrate animation using shapes.

GRAPHING DEMO

This graphs a sin wave on the hi-res screen.

PEN DEMO

This is a simple drawing demonstration on the hi-res screen and requires the use of a light pen.

ONERR DEMO

The use of the ONERR command is shown in this program.

HIRES DEMO

Another demonstration of hi-res graphics.

SOUND DEMO

The sound commands are used to make some sounds.

SPRITE DEMO

This program recquires the use of paddles in port 1, and will move a sprite around on the text screen.

FILECABINET

This is a famous Apple public domain database, and should be saved on a seperate disk, as it requires the use of data files. It is examined in Appendix 8.

ART TRIANGLE

This is a public domain Applesoft program, that uses hi-res graphics.

CONVERT

This is a boot program for the CONVERT 1.0 program. It cannot be used in ApSoft-64, but must be loaded and run in normal C64 BASIC

TVPATTERN

Another public domain Applesoft program.

HIDDENLINES

This is another public domain Applesoft program.

CHECKBOOK BAL

This public domain Applesoft program will help balance a checkbook.

SHAPES $8000

This is a binary file that loads at $8000 (decimal 32768). It is a shape table that contains text shapes. Shapes 1 through 26 are lower case alphabet characters, 27-32 are 'spaces', and 33 - 95 are normal ASCII characters, the alphabetic characters being upper case. Loaded from an Apple.

MACH.CHARSET—C64 CHARSET

These two files are character sets to be used with the secondary text page. Loaded by 'PG2 CHAR DEMO'.

BADLINE DUMP

This program is to be used with CONVERT to read the file that is created for bad lines when converting an Applesoft program.

ART XMAS SCENE

An APPLE public domain program, it demonstrates the use of LO-RES graphics.

SIMPLE TERMINAL

This is a simple BASIC terminal program that will assist in downloading APPLE programs. It will save all incoming text in a SEQ file.

SPRITE EDITOR

A program to edit sprites

SPRITE INSTR

Instructions for the sprite editor

SINGLE FILE COPY

Copies files from one disk to another.

SHAPE CREATE

An editor to create shape tables with.

HI-RES SAVE

Program to save hi-res screens for re-loading.

SPRITE BOOT

The program which loads the sprite editor.

TINY AID

Program to search, replace, append programs in memory.

SPRITE GAME/DEMO

A simple game demonstrating the sprite commands.

APPENDIX J

USING THE CONVERT PROGRAM

This program will read a sequential file containing an APPLESOFT program and will convert it into a program file that can be loaded, listed and run. In doing so it will also create a file containing the numbers of lines containing commands that may be a problem.

CONVERT will display a menu giving a choice to either convert a sequential file, catalog a disk, or exit to BASIC. Prior to running this program, the program to convert should have been downloaded from an Apple into a sequential file using either a modem or an RS232 adapter connected to an APPLE, and a suitable communications program in the Commodore 64, or the SIMPLE TERMINAL program included on the ApSoft-64 disk.

The proper method for downloading directly from the Apple to the Commodore using an RS232 adapter is to load a communications program that allows the saving of text files in a sequential file, then set the parity to NO parity, the number of data bits to 8, and the number of stop bits to one. This is the standard configuration of most Apple serial cards and the Apple IIc serial ports. At this point the communications cable should be connected to the Apple. On the Apple, boot the disk that contains the program(s) that are to be downloaded, and LOAD the file that is to be transmitted. Next, type in POKE 33,30 <return> and PR#<slot # of the communitions> <return>. When ready to receive the file, type in LIST, and THEN hit the return key on the Apple ONLY as soon as the Commodore is ready to receive the file. The POKE command will set up the Apple to list each program line with out any returns, and the PR# will turn on the communictions card. Keep in mind that any program line that contains a REM can have some strange control characters, and CONVERT will remove them, although occasionally text following the REM may be lost.

After the files have been transmitted and received, the CONVERT program can be RUN to convert the program. If any problems are encountered in this process, refer to the manual received with the RS232 adapter, or your MODEM.

Using a modem is similar, except getting access to an Apple bulletin board is the means to download programs. In this case follow the directions on accessing the bulletin board to download any files.

During the actual program conversion, the lines will list as they are crunched and converted. When each line is listed, it will be crunched and placed in memory. When the entire program is listed, a prompt will appear to place a disk in the drive so the file can be saved. This disk must be formated. A file will also be saved that contains lines that must be examined because of commands that will cause possible problems. The program is saved prefixed with a 'P.' and the questionable lines are saved in a sequential file prefixed by a 'B.', and can be read and printed out by the program called BADLINES DUMP on the program disk. If the B.file is not on the disk the P. file is saved on, no bad lines were found. The program should still be checked for possible problems, however.

The section in this manual on program conversion will be of assistance when the program is listed and the problem lines examined and changed. Keep in mind that the Apple allows lines up to 255 characters, and any lines found over 80 characters must be divided into two or more lines, making sure that the integrity of the program is maintained. Any lines that list over 255 characters will have all the spaces removed to keep them under 255 characters. Listing the program on a printer will be of assistance also, mapping out any subroutines and other lines that may have to be altered.

The program CONVERT should be loaded from Commodore BASIC, and will return to Commodore BASIC when done.

Please note that CONVERT will check only for the following

suspicious commands:

POKE PEEK & SHLOAD GET PR# IN#
CALL ONERR RECALL STORE WAIT USR SCRN(

Also, the following commands are not crunched and the program line will have to be retyped, deleting the command (it is not supported):

IN# RECALL STORE USR SCRN(

INCOMPATIBILITIES BETWEEN APPLESOFT AND APSOFT-64

Although ApSoft-64 is highly compatible with the Applesoft format for the Apple, there are a few areas of incompatibility. Although most programs lend themselves to easy conversion, the longer and more complex programs (especially poorly written or documented ones) can present problems. Many hardware functions on the Apple are not supported, such as the various slots open in the Apple for peripheral cards, or special locations used to access the Apple's I/O. Disk and file handling can create special problems and Commodore does not support scrolling windows. Certainly, if a program does not work correctly on the Apple, it will not be fixed by converting it to ApSoft-64.

The INPUT command can cause problems in some situations. Applesoft will zero out any variable asked for in the input command if only a [RETURN] is entered. The Commodore, on the other hand, leaves the variable unchanged from the value it had before the INPUT. A problem also exists with an INPUT statement that has a prompt that extends more than 40 characterers. If the prompt (i.e. INPUT "prompt" ; A$) is longer than 40 characters, the operating system will append the prompt to the input you enter. This is actually a bug in Commodore's operating system.

Any program that contains any machine language may be impossible to convert, unless the purpose of the machine language is apparent, and it can be added without any memory conflicts. In any case, a knowledge of both computers would be helpful.

A program called CONVERT is included on the disk to assist you in converting Applesoft programs to ApSoft-64. It will read a sequential file containing an Applesoft program downloaded from an Apple and convert it into a format that ApSoft-64 can use. It will read the file, convert it, then save it on a disk along with a file that can be printed out with BADLINE DUMP that will list any program lines that might need to be looked at.

The following information is provided as a reference for those interested in converting Apple software. If extensive information about Applesoft and Apple DOS is needed, it may be necessary to refer to the Apple manuals for further information. Likewise, much more information on Commodore DOS and associated commands can be found in the Commodore manuals.

DOS COMMANDS

Applesoft allows the use of the DOS commands only in PRINT statements, preceded by a 'control D' (seen sometimes as D$ or CHR$(4)). ApSoft-64 does not support this format. DOS commands in ApSoft-64 are used like any other BASIC commands. See the previous chapters for the details.

Although Applesoft allows many files to be open simultaneously, only one file can be read from or written to at one time. On the Commodore, only 10 files may be open at one time. Also, Applesoft DOS treats all print statements after an open-to-write as a write to the file, not to the screen. The same thing is true with the INPUT or GET statements following an open-to-read. This mode is ended by a print statement containing a CLOSE command.

Apple DOS allows several other commands that may give problems. One of these is the POSITION command, which will appear in a PRINT CHR$(4) format also. It allows the program to read from a sequential file starting at any point in the file. This command in the program will occur prior to any READ or WRITE statement. Some other programs may even contain a WRITE <name of file>, B10 command, and this command will overwrite the named file starting at the 10th byte of the file.

Having a good knowledge of both computers will be most important in converting any program that uses much file handling.

POKES, PEEKS AND CALLS

Peeks and pokes have many uses on the Apple: for example, setting the text window, reading the keyboard, switching graphics modes, toggling the speaker, and so on. The Commodore 64 does not support some of these uses and enhances some of the others. Following is a list of some of these.

TEXT WINDOWS

Text windows on the Commodore 64 are not supported, and the following locations, if referenced in an Applesoft program, will require a significant 'rewrite' of the program. The rewrite may require analyzing the use of the screen and the format of the screen, and re-aranging print statements so that the text appears the same on the Commodore screen.

The following addresses, if accessed in an Apple program by either a PEEK or a POKE, will mean text windows are being used and may require some rewriting of the program: (All locations are in decimal)

LOCATION	USE	COMMODORE EQUIVALENT
32	Left margin	NONE
33	Right margin	NONE
34	Top margin	NONE
35	Bottom margin	NONE

PEEKS, POKES AND CALLS

LOCATION	USE	ApSoft-64 EQUIVALENT
CALL-936	Clear all text in text window.	HOME; CALL/SYS 59592
CALL-958	Clears all text from cursor on.	NONE
CALL-868	Clears current line from cursor on.	NONE
CALL-922	Prints a line feed.	PRINT CHR$(10); CALL/SYS58692 OR PRINT"<CRSR DN>";
CALL-756	Waits for key press.	WAIT 197,190
PEEK(-16384)	Reads keyboard value	WAIT 197,190: GETA$:X=ASC(A$)+128 X has the same value as the PEEK
PEEK(-16368)	Resets keyboard strobe	NOT NEEDED
PEEK(36)	Reads in position of cursor in text window (0 - 39)	PEEK(211) (0 - 79)
POKE(36)	Sets position of cursor in text window (0 - 39)	POKE(211) (0 - 39)

PEEK(37)	Reads in vert. pos. of cursor in text window (0 - 23)	PEEK(214) (0 - 24)
POKE(37)	sets vert. pos. of cursor in text window (0 - 23)	POKE(214) (0 - 24)
CALL -912	Issues a line feed.	CALL/SYS 59626 PRINT CHR$(10) PRINT "<crsr down>"

This command will scroll the screen up one line and does not affect the cursor position.

POKE(232)	Sets lo order address of shape table	POKE 53017
POKE (233)	Sets hi order address of shape table	POKE 53018

COMMANDS AFFECTING GRAPHICS

ADDRESS ACTION

-16299.....TURN ON TEXT OR GRAPHICS PAGE 2
-16300.....TURN ON TEXT OR GRAPHICS PAGE 1
-16301.....TURN OFF SPLIT SCREEN
-16302.....TURN ON SPLIT SCREEN
-16303.....TURN ON TEXT/WHOLE SCREEN
-16304.....TURN ON GRAPHICS/WHOLE SCREEN

NOTE: All of the above locations can be either POKEd or PEEKed and will have a simalar effect as with Applesoft. Also, although the following two locations will not yield errors, they are not acted upon at all: -16298 and -16297

COMMANDS DEALING WITH GAME CONTROLS

LOCATION (PEEK/POKE)	RESULT	COMMODORE
-16336	Produce click on speaker	NONE- use PLAY/ SOUND commands
-16287	Toggles cassette output once.	NONE

READING GAME CONTROLS

COMMAND	RESULT	COMMODORE
X=PEEK(-16287)	Read pushbutton for game controller #0	X=(PDL(0))AND8
X=PEEK(-16286)	Read pushbutton for game controller #1	X=(PDL(1))AND8
X=PEEK(-16285)	Read pushbutton for game controller #2	X=(PDL(0))AND8
NONE	Read pushbutton for game controller #3	X=(PDL(1))AND8

DOS COMMAND CONVERSION

FOR THIS	USE THIS
PRINTCHR$(4)"DELETE PRGM NAME"	OPEN15,8,15,"S0:PRGM NAME" :CLOSE15
PRINTCHR$(4)"RUN PRGM NAME"	RUN PRGM NAME
PRINTCHR$(4)"BLOAD PRGM NAME"	BLOAD PRGM NAME
PRINTCHR$(4)"LOAD PRGM NAME"	LOAD PRGM NAME
PRINTCHR$(4)"CATALOG"	CATALOG
PRINTCHR$(4)"OPEN FILENAME" PRINTCHR$(4)"WRITE FILENAME"	OPEN 2,8,2,"FILENAME,,SEQ,W"

NOTE: File output can be accomplished on the Commodore by following the OPEN command with either CMD2 and normal PRINT statements or PRINT#2 statements without the CMD2.

PRINTCHR$(4)"READ FILENAME"	OPEN 2,8,2,"FILENAME,SEQ,R" NOTE:Use INPUT#2 statements or GET#2 statements to retrieve information.
PRINTCHR$(4)"CLOSE FILENAME"	CLOSE2
PRINTCHR$(4)"APPEND FILENAME" PRINTCHR$(4)"WRITE name	OPEN 2,8,2,"name,SEQ,A":CMD2 NOTE: The OPEN command can be followed by either CMD2 and normal PRINT statements or just PRINT#2 statements.
PRINTCHR$(4)"OPEN FILENAME,L200"	OPEN2,8,2,"FILENAME,L"+

	CHR$(RECORD LENGTH)
PRINTCHR$(4)"WRITE FILENAME,R2"	PRINT#15,"P"CHR$(CHANNEL #); CHR$(RECORD# LO)CHR$(RECORD# HI)

OTHER AREAS OF POSSIBLE INCOMPATIBILITY

The TAB command functions differently in some situations. In general, the tab will work the same on both machines, except when printing on a printer. The Apple will automatically tab from the beginning of a printed line and the Commodore will tab from the last printed position. If the tab position is before the last printed position, a line feed will be generated on the Commodore and not on the Apple. It will also be important to be on the lookout for a poke to the location 37, because some Apple printer cards will recognize that as a tab and act accordingly.

Logical expressions generate differing values for a true condition on the two systems. For example, B=(A>1) will give B a value of either -1 or 0 on the Commodore and a 1 or a 0 on the Apple. An easy fix is to use ABS of the logical expression: i.e. S*(A>4) should become S*ABS(A>4)

An additional problem may occur with the use of ONERR GOTO statements. This statement will not check the disk drive error channel for a file not found if an open statement is being used. Any error that must be read from the disk drive must happen in the program after the open statement. Also, the ONERR statement will be accessed if ANY error occurs, so if your program appears to be crashing and an ONERR statement is being used, check for SYNTAX or other errors in the program that are not expected.

The Apple's GET statement will wait until a key is pressed before returning a value, while the Commodore will not. A simple solution to this problem is to replace each Apple GET statement with the following: WAIT 197,190:GETA$. This will wait until a key is

depressed before performing the GET.

The use of lo-res commands will have to be watched also. The APPLE allows vertical positions from 0 through 47, and ApSoft-64 allows 0 through 24. An easy way of solving this problem and maintaining a similar appearing screen is to divide all of the vertical positions by two and drop the remainder. The statements to watch for would be VLIN X,Y at Z (change the X and Y); HLIN X,Y at Z (change the Z); and PLOT X,Y, (change the Y).

Remember also that APPLE programs can use variable names that are illegal on the Commodore 64, such as TI; TI$; IN; and ST. These variable names will have to be changed.

Some of the CALLs which have no direct equivalent in ApSoft-64 may be simulated with a short BASIC subroutine. For example, a CALL -958, which clears the screen from the cursor position on, can be simulated with the following:

```
1000 X=PEEK(211):Y=PEEK(214) :REM GET CURSOR X,Y POSITION
1010 FORX1=YTO40:?" ";:NEXT :REM CLEAR TO END OF LINE CURSOR IS ON
1020 FORX1=XTO24:?"<40 SPACES>";:NEXT :REM CLEAR TO END OF SCREEN
1030 VTABX:HTABY:RETURN :REM SET CURSOR POSITION
```

Some APPLE programs use the ESCAPE key to exit the program, move to a different menu, etc. This key is not on the Commodore 64 keyboard, and another key should be substituted. The ASCII code for the ESCAPE key is (decimal) 27.

Several APPLE commands are not implemented at all. These are: FLASH STORE RECALL & SCRN(. Also, a number of APPLE DOS commands are not supported. These are:

INIT RENAME DELETE LOCK UNLOCK MON NOMON MAXFILES.
FP INT IN# CHAIN POSITION EXEC BSAVE BRUN.

Remember that programs which don't work on the Apple will not work on the Commodore. CONVERT will not correct logic errors!